



Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

A BRIEF INTRODUCTION TO COQ

The Coq Proof Assistant

3. Februar 2020

Tanja Almeroth

Studienbereich DCSM
Hochschule **RheinMain**



TABLE OF CONTENTS

1. proof assistants
2. functional programming in coq
3. an example
4. applications

PROOF ASSISTANTS

THE COQ PROOF ASSISTANT

usage and applicability of the Coq proof assistant

- developed since 1983
- community in the industry and research
- basic concepts of logic
- functional programming



Abbildung: the rooster

LOGIC

"As a matter of fact, logic has turned out to be significantly more effective in computer science than it has been in mathematics." [1]

LOGIC

Reliability in software is amplified by the costs of bugs by insecurity up to multiple levels

- basic tools from logic
- precise claims about programs
- functional programming methods of programming and logical reasoning about programs

FUNCTIONAL PROGRAMMING IN COQ

SYSTEM REQUIREMENTS

Coq runs on Linux, macOS and Windows.

- Proof General: an Emacs based IDE
- CoqIDE is a simple stand-alone IDE
- coquille: a vim plug-in

SYSTEM REQUIREMENTS

Coq runs on Linux, macOS and Windows.

- Proof General: an Emacs based IDE
- CoqIDE is a simple stand-alone IDE
- **coquille**: a vim plug-in

FUNCTIONAL PROGRAMMING

Coq's functional programming language *Gallina's* sub-types

1. vernacular language (top-level interaction)

e.g.: `Theorem`, `Proof`, `Qed`.

2. tactics language:

e.g.: `intros`, `exact`

3. the *language of Coq-terms*:

e.g.: `for` all `A`: `Prop`, $A \rightarrow A$.

AN EXAMPLE

AN EXAMPLE

```
1 Inductive day : Type :=  
2   | monday  
3   | tuesday  
4   | wednesday  
5   | thursday  
6   | friday  
7   | saturday  
8   | sunday.
```

Listing 1: definition of a datatype called `day`

AN EXAMPLE

```
1 Definition next_weekday (d:day) : day :=  
2   match d with  
3   | monday    ⇒ tuesday  
4   | tuesday   ⇒ wednesday  
5   | wednesday ⇒ thursday  
6   | thursday  ⇒ friday  
7   | friday    ⇒ monday  
8   | saturday  ⇒ monday  
9   | sunday    ⇒ monday  
10  end.
```

Listing 2: declaration of a function on `day`

AN EXAMPLE

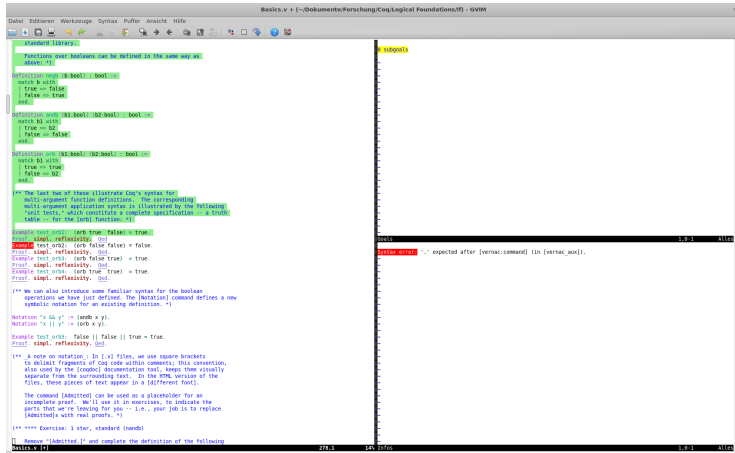
```
1 Compute (next_weekday friday).
```

Listing 3: call `compute`

```
1 = monday: day
```

Listing 4: output

USER INTERFACE



```
standard library.
Functions over booleans can be defined in the same way as:
above: *)

Definition andb : A bool -> bool :=
match b with
| True => false
| False => true
end.

Definition orb : A1 bool A2 bool -> bool :=
match b1 with
| True => b2
| False => false
end.

Definition orb2 : (A1 bool A2 bool) -> bool :=
match b1 with
| True => true
| False => b2
end.

(** The last two of these illustrate Coq's syntax for
multi-argument function definitions. The corresponding
multi-argument application syntax is illustrated by the following
"unit tests", which constitute a complete specification ... A truth
table ... for the [orb] function: *)

Example test_orb1 : (orb true false) = true.
Proof. simpl. reflexivity. Qed.
Example test_orb2 : (orb false false) = false.
Proof. simpl. reflexivity. Qed.
Example test_orb3 : (orb false true) = true.
Proof. simpl. reflexivity. Qed.
Example test_orb4 : (orb true true) = true.
Proof. simpl. reflexivity. Qed.

(** We can also introduce some familiar syntax for the boolean
operations we have just defined. The [Notation] command defines a new
symbolic notation for an existing definition. *)

Notation "x && y" := (andb x y).
Notation "x || y" := (orb x y).

Example test_orb5 : false || false || true = true.
Proof. simpl. reflexivity. Qed.

(** A note on notation. In [v] files, we use square brackets
to delimit fragments of Coq code within comments; this convention,
also used by the [coqdoc] documentation tool, keeps them visually
separate from the surrounding text. In the HTML version of the
files, these pieces of text appear in a different font.

The command [Admitted] can be used as a placeholder for an
incomplete proof. We'll use it in exercises, to indicate the
parts that we're leaving for you -- i.e., your job is to replace
[Admitted]s with real proofs. *)

(** **** Exercise: 1 star, standard (randb)

Remove [Admitted] and complete the definition of the following
```

Abbildung: Coq interface in gvim using coquille.

APPLICATIONS

APPLICATIONS

A platform for *modelling programming languages* and an environment for *formally certifying software and hardware*

- PROSA a felxible open-source foundation for formally proven schedulability analysis. PROSA uses the Coq proof assistant and the SSREFLECT extension library
- ...

LITERATURVERZEICHNIS



B. C. Pierce and A. A. de Amorim and C. Casinghino and
M. Gaboardi and M. Greenberg and C. Hrițcu and V. Sjöberg
and B. Yorgey

»Software Foundations, Logical foundations, Volume 1«
<https://softwarefoundations.cis.upenn.edu/current/lf-current/index.html>, 2019



»The Coq Proof Assistant«
<https://coq.inria.fr> , 2019-01-09



»Prosa - ECRTS'16 Artifact Evaluation«
<https://prosa.mpi-sws.org/releases/v0.1/artifact/>,
2020-01-09



F. Cerqueira and F. Stutz, and B. Brandenburg
»Prosa: A Case for Readable Mechanized Schedulability